DOCUMENT RESUME

ED 066 862 EM 010 095

AUTHOR Roid, Gale H.
TITLE User's Guide to "MULE"; McGill University Language for Education. A Computer-Assisted Instruction Author Language.
INSTITUTION McGill Univ., Montreal (Quebec).
PUB DATE 72
NOTE 8p.

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS College Students; *Computer Assisted Instruction; Computer Programs; Guides; *Programming Languages
IDENTIFIERS McGill University; *MULE computer language

ABSTRACT

A computer-assisted instruction (CAI) author language and operating system is available for use by McGill instructors on the university's IBM 360/65 RAX Time-Sharing System. Instructors can use this system to prepare lessons which allow the computer and a student to "converse" in natural language. The instructor prepares a lesson by coding text material, questions, and answers in a special CAI language. The coded lesson is prepared for input to the MULE compiler. Once the lesson has been placed in a disk file it can be called upon from any remote terminal connected to the RAX system by a student who then proceeds through an instructional dialogue with the executing program. Basic parts of the MULE language are explained here, including the statement form, label field, operation code field, operation codes for display statements, operation codes for response processing statements, and operation codes for control and accounting statements. Student's times, responses, and scores are automatically recorded on a permanent file. (JK)

User's Guide to "MULE"

McGill University Language for Education:

A Computer-Assisted Instruction Author Language

Gale H. Reid
McGill University

SECTION I: Introduction

A computer-assisted instruction (CAI) author language and operating system is available for use by McGill instructors on the university's IBM 360/65 RAX Time-Sharing System. Instructors can use this system to prepare lessons which allow the computer and a student to "converse" in natural language.

The instructor prepares a lesson by coding text material, questions and answers in a special CAI language. The coded lesson is prepared for input to the MULE compiler. Once the lesson has been placed in a disk file it can be called upon from any remote terminal connected to the RAX system by a student who then proceeds through an instructional dialogue with the executing program.

SECTION II: The MULE Author Language

Lessons are prepared for computer input by coding them in the MULE language. "Coding" means writing lesson material in a special form involving many individual statements. Each statement is made up of at least three parts: 1) a label or number for the statement,

2) an operation code, and 3) written material to be read from or typed onto a computer terminal. Each statement must be less than 80 characters in length, in order to conform to IBM card restrictions, as illustrated below:

Statement Form:

Information:	Label	Operation Code	Text
Card Column:	1 2 3 4	5 6 7 8	9.....80

The Label Field

Labels for each statement are optional. The main purpose of labels is to identify points in a coded lesson to which a student is to be branched (see description of the "GTO" operation code). Labels consist of from 1 to 4 alphabetic or numeric characters. Each label must start in the first card column (although leading blanks can be used as "characters").

Operation Code Field

Operation codes are three or four-letter symbols which indicate to the MULE compiler the operation that is to be performed at that point in the lesson. These codes are typed into positions (columns) 5, 6, 7, and 8 of each statement. There are four basic types of statements in the MULE language, each with its own set of operation codes, 1) Display statements that ask for information to be typed out on the student's teletype, 2) Response Processing statements that call for and inspect a response from the student, 3) Accounting statements that allow records to be kept of types and frequencies of student responses, and 4) Control statements which direct the sequence in which lesson statements are executed

(viz., "GTO" branching statements), or signal the end of a lesson.

Each Operation code will be described in turn.

Operation codes for Display Statements

1. "PRO" PROblem. The information typed in cols. 9-80 of a PRO statement are written out on the teletype. PRO is meant to be used as the first line of text written on the teletype for each discrete problem, frame, question or unit. Triple spacing above the line is done automatically. On the permanent record of student responses, the responses corresponding to each PRO statement are written, each PRO being numbered from the first to the nth (see SECTION III on accounting).

2. "PRE" PREsent lesson material (text). The information typed in columns 9-80 of a PRE statement is written out on the teletype, with triple above it. A PRE statement is used for introductory comments or directive.

3. "XXX" Continuation of text or information. The characters in columns 9-80 of an XXX statement are typed out on the teletype with single spacing.

   XXX can also be used to type messages to the student after a response has been processed (see XXX below).

4. "SHO" SHOw the contents of a counter. Can be used to display student's score, or other accounting information. See description below.

Operation Codes for Response Processing Statements

1. "ANS" ANSwer. When an ANS statement is encountered by the MULE system during execution, the teletype is set up to receive a response from the student. ANS causes a "?" to be typed out on the teletype and then causes the system to read in one line of information from the student.

2. "GUD" "Good" Response. Causes the student's response to be compared with the characters in columns 9-80 of the GUD statement. An exact, character for character match is necessary. For example, the statement:

		G U D		IBM CARD
Column	1 2 3 4	5 6 7	8	9.....80

would cause the student's response to be compared with "IBM CARD". (More than one GUD can be used for any one ANS).

3. "BAD" BAD response. Causes the student's response to be compared with the characters in columns 9-80 of the BAD statement. Exact match necessary. Equivalent to a GUD statement in function, but difference in name ("Good" vs. "Bad") helps instructor write lesson clearly. (More than one BAD can be used for any one ANS.)

4. "KWG" (Key Word Good)

"KWB" (Key Word Bad)

These codes direct the computer to search the student's response for character strings ("Key words"). Their format is as follows:

		K W G		/W1/,/W2/,/W3/;ORDER=x;MATCH=n
column	1 2 3 4	5 6 7	8	9.....80

"W1", "W2", and "W3" are key words to be searched for in the student's response. Any characters except comma (,), slash (/), and semi-colon (;) may be used in the key words. The whole instruction must fit on one card, although several may be used for any one ANS.

The variable ORDER indicates whether the order of the key words is important. "ORDER=" may be abbreviated to "O=" or "ORD=" followed by a YES (or Y) or NO (or N). ORDER=YES implies that the order of key words given on the KWG or KWB card must be found for a positive match to occur; ORDER=NO that order is not important.

The variable MATCH indicates the number of key words which must be found in order for a positive match to occur. "MATCH=" may be abbreviated to "M=" or "MAT=" followed by a number 0 to 9. The number specifies how many key words must be identified. (If "MATCH=24" is specified erroneously, the program assumes "MATCH=2" was specified).

If only one key word is to be searched for, then the instruction may be abbreviated to "KWG /W1/". In all other cases ORDER and MATCH must be specified. Spacing is relatively unimportant as is the order of ORDER and MATCH. Only if these parameters are NOT abbreviated can the verbs "IS" and "ARE" be used to replace the equal sign (=).

Some examples follow:

```
KWG /SEMI/,/CONDUCTOR/;MATCH=2;O=Y

KWG /1/,/2/,/4/;M=1;ORDER IS NO

KWB /TWO/,/2/;O=N;M=1
```

5. "UNX" UNeXpected response. If student's response does not match a GUD or BAD, an unexpected response can be processed with this statement. An integer number from 1 to 9 must accompany UNX, in column 9, i.e.,

		U N X		n
Column	1 2 3 4	5 6 7	8	9.....80

where "n" is an integer number indicating the number of times an unexpected response will be allowed for any one ANS, before the student is moved on to the next frame (the next PRE).

6. "XXX" An XXX continuation statement can be used to follow GUD, BAD, and UNX statements, in which case the information in columns 9-80 of the XXX statement are typed on the student's teletype after a positive match (or "no match" in the case of UNX). This makes it possible for verbal feedback, instructions or other comments to be given to the student immediately after his response.

Example of response processing:

```
Col.:    1234567 8 9 ..... .....

         PRO  HOW ARE YOU TODAY?
B1       ANS
         KWG  /FINE/,/OK/,/NOT BAD/;O=N;M=1
         XXX  THAT'S GOOD
         KWB  /BAD/,/TERRIBLE/;O=N;M=1
         XXX  TOO BAD
         UNX  1
         XXX  SORRY, I DIDN'T READ YOU
         GTO  B1
         PRO  WELL, LET'S GET ON WITH THE LESSON
          :
          .
         END
```

Operation Codes for Control Statements

1. "GTO" Go TO. A branching statement causes the lesson to be continued at the statement given by the label in columns 9, 10, 11 of the GTO statement. Used to break the sequence of statement execution.

If a GTO is used after a GUD, BAD, or UNX, the branch is made only if a positive match is made (or "no match" is the case of UNX). See example of response processing above.

2. "END" — END of lesson. Signals the MULE compiler of the end of lesson code. Used only once per lesson as the very last statement.

Operation Codes for Accounting Statements (See also SECTION III below)

1. "ADD" — ADD to a counter for record keeping, scoring, etc. Up to 200 counters can be used to accumulate scores or tallies during lesson.

Information starting in column 9 of the ADD statement is used to indicate 1) whether a positive or negative quantity is to be added to a counter, 2) the quantity to be added to the counter, and 3)the number of the counter. Each of these three elements is separated by a comma, e.g.:

```
ADD   +, 1, #5

ADD   —, 2, #101
```

Also, two counters can be operated on, e.g.:

```
ADD     , #2, #3
```

(which would add the contents of counters #2 and #3).

2. "SHO" — SHOw the contents of a counter. Specified counter contents are written on the teletype along with verbal comments if desired.

The SHO statement has three possible forms:

1) SHO comment; #m; more comment

For example:

```
SHO    YOU HAVE GOTTEN; #16; CORRECT SO FAR
```

2) SHO comment; #m

For example:

```
SHO    NUMBER RIGHT=; #2
```

3) SHO ; #m

For example:

SHO ; #151

3. "TIME" TIME (typed in columns 5, 6, 7 & 8) used once at the beginning of a MULE lesson, starts a timing routine that computes the total amount of elapsed time the students take to complete the lesson.

SECTION III: Automatic Accounting and Student Record File

1. Automatic Accounting Function

MULE incorporates the function of the ADD instruction automatically for every GUD or KWG, BAD or KWG, and UNX response. Counters 0, 1, and 2 respectively are reserved for counting the number of UNX's, GUD's or KWG's, and BAD's or KGB's. For example, if you wished to tell a student how he was doing on a lesson you might use the following SHO statements:

```
SHO YOU HAVE; #1; CORRECT SO FAR
SHO YOU HAVE MISSED: #2
SHO AND; #0; WERE UNEXPECTED
```

2. Permanent File for Student Records

The responses, time, and scores for each student are automatically recorded on a permanent file (supplied by the user) for use by the instructor for accounting purposes. The MULE compiler constructs an implicit numbering of the PRO's in a lesson from "1" to n, the total number of PRO's. The responses of each student for each PRO are written on a permanent record file with their associated number and GUD, BAD or UNX classification. Below is a sample of a student record:

```
        1.  GUD *** RESPONSE *** COLLEEN HORAN
        2.  GUD *** RESPONSE *** D
        3.  GUD *** RESPONSE *** POSITIVE
        4.  UNX *** RESPONSE *** RECOVERY
        5.  BAD *** RESPONSE *** B
*TOTAL GUDS = 3
 TOTAL BADS = 1
 TOTAL UNXS = 1
 TOTAL ELAPSED TIME = 3.48 MINS
*$$* DATE OF FILE ENTRY:  15 JUL 70
***  TIME OF DAY AT COMPLETION:  10.31.05
```

To obtain a printout of your permanent file (each page showing a record such as that above for each student's use of a lesson), you can use a program called DSPLAY which can be run from a teletype terminal. Another program, MULER, can be run through batch processing to have the file printed on a high-speed printer. A program called RENEWP is used to erase a permanent file which has been printed, is filled, and needs to be reused.